

**UE22CS352B - Object Oriented Analysis & Design**

Mini Project Report

TOURISM MANAGEMENT SYSTEM

***Submitted by:***

***Nandakumar.T PES1UG22CS375***

***P.Ashish- PES1UG22CS404***

***PRAJWAL N.P PES1UG22CS423***

***PAVAN T.R PES1UG22CS411***

Semester-6 Section-G

# Facultly Name:DR.BHARGAVI MOKASHI

**January - May 2025**

##### DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING

##### PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013) 100ft Ring Road, Bengaluru – 560 085, Karnataka, India

##### Problem Statement: TOURISM MANAGEMENT SYSTEM

**Key Features:**

🡪The main objective of the Tourism Management System is to manage the details of Customer, Hotel Booking, Cancellation and Tourism places. It manages all the information about Users, Hotel, Packages etc. The project is totally built at administrative end and thus only the administrator is guaranteed the access to the backend database.

🡪The purpose of this project is to build an application program to reduce the manual work for managing Tourists, Booking, Places etc.

🡪This application will help in accessing the information related to the travel to the particular destination with great ease. The users can track the information related to their tours with great ease through this application. The travel agency information can also be obtained through this application.

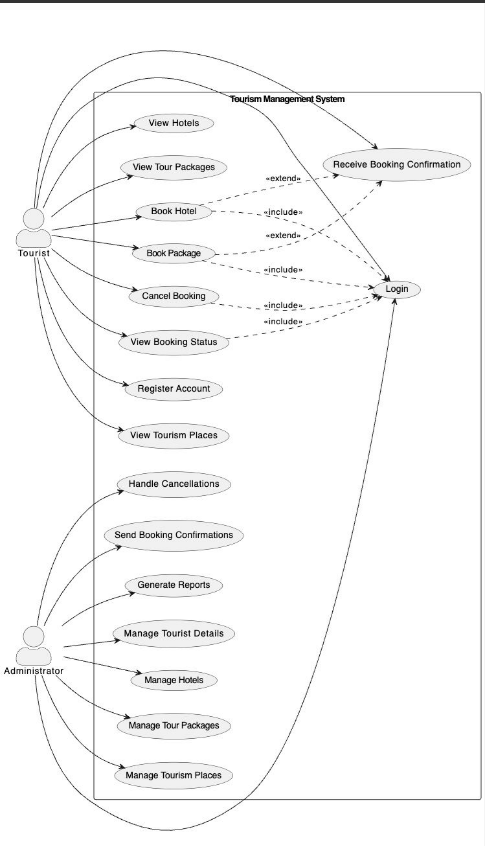
🡪 Through this system, the propose system is highly automated and makes the travelling activities much easier and flexible. The user can get the very right information at the very right time. This system will include all the necessary fields which are required during online reservation time. This system will be easy to use and can be used by any person. The basic idea behind this project is to save data in a central database which can be accessed by any authorize person to get information and saves time and burden which are being faced by their customers.

🡪Administrator can access and modify the information stored in the database of this system, this includes adding and updating of details, and it will give accurate information and simplifies manual work and also it minimizes the documentation related work. Provides up to date information. Finally booking confirmation notification will be send to the users.

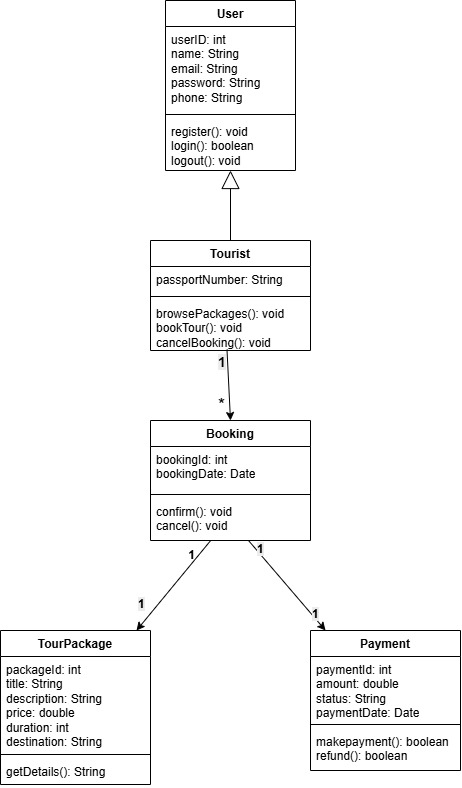
🡪Tourists can register by providing personal details, make new reservation and book only one hotel and package and can make cancellation.

## Models:

Use Case Diagram:



**Class Diagram:**

****

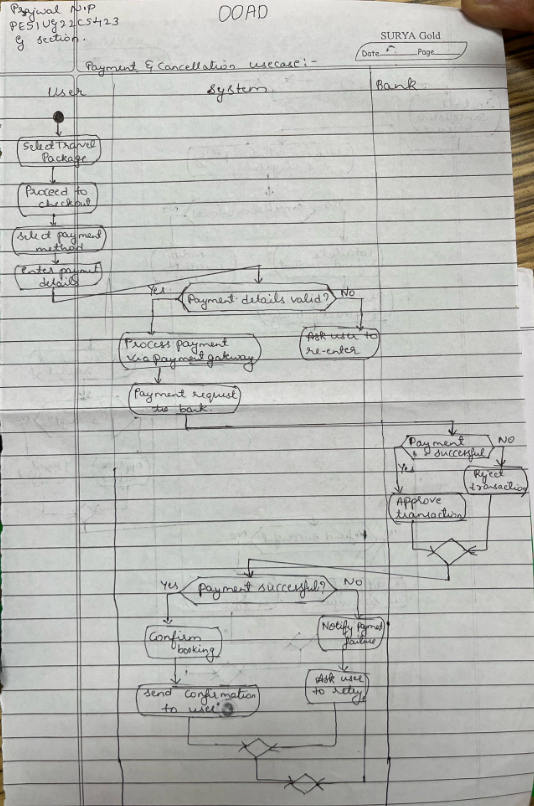
# State Diagram:

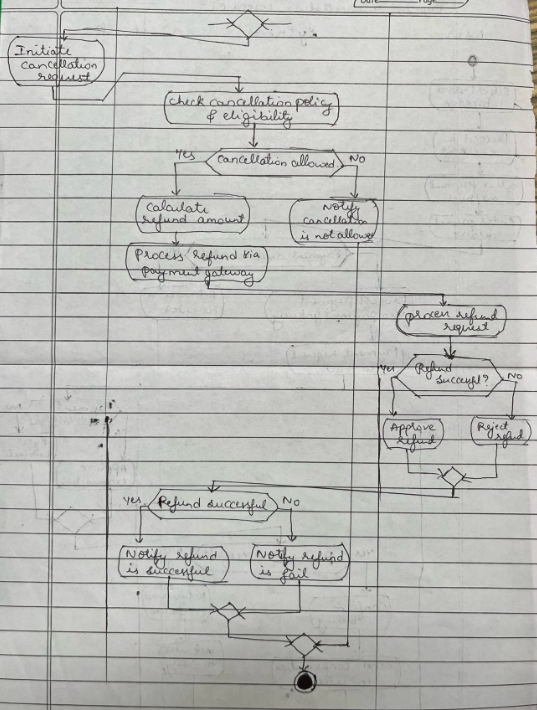
# 

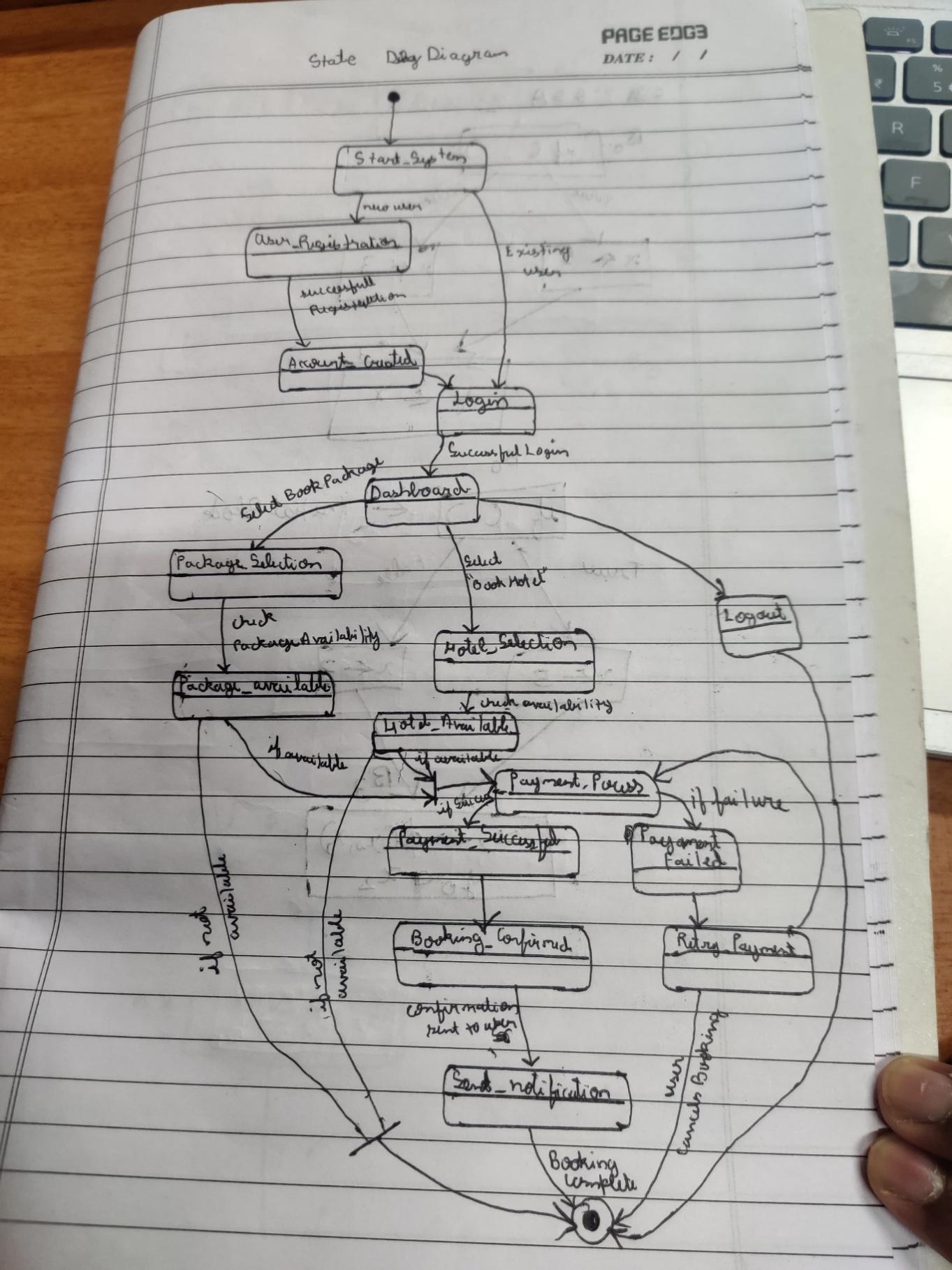
**Activity Diagrams:**

1. Major Usecase

* Hotel Booking:
* Package booking
* User Management
* Payment and cancellation







Architecture Patterns, Design Principles, and Design Patterns:

# Architecture Patterns

# Layered Architecture (2-tier)

# The travel management application follows a basic 2-tier Layered Architecture:

# Presentation Layer (Java Swing UI):

# Classes like UpdateCustomer, ViewCustomer, ViewPackage, ViewBookedHotel, etc., handle GUI rendering and user interactions.

# Data Access Layer (via JDBC):

# Each class performs SQL queries directly using JDBC to connect to the database and retrieve or update data.

# There is no separate business logic or service layer, so the code mixes data access and UI logic in the same class.

# Design Principles

|  |
| --- |
| Single Responsibility Principle (SRP) |

|  |
| --- |
| Partially followed. Each form like UpdateCustomer or ViewCustomer has a single responsibility (e.g., updating or viewing data). But database and UI logic are not separated, which still violates SRP. |

# Design Patterns

# Design Patterns USED

# These are the actual design patterns present in your code:

# 1. Prototype of MVC

# Although not fully implemented, your design reflects a primitive MVC-like structure:

# View: Classes like ViewCustomer, UpdateCustomer represent GUI.

# Controller + Model: There is no separation between controller and model. JDBC and logic are embedded inside the View classes.

# 2. Builder Pattern (implicitly via Swing Layouts)

# We have used builder-style UI creation, where Swing components are built step by step and then added to the frame:

# JLabel label = new JLabel("Username");

# label.setBounds(30, 50, 100, 30);

# add(label);

# While this isn’t the formal Builder Pattern, it mimics the idea of incrementally assembling an object (the UI).

### Github link to the Codebase:

<https://github.com/neeharika-anand/grocery-store>

# Screenshots:

# 

## Login Page :

## 

## Signup Page:

## 

Individual contributions of the team members:

|  |  |
| --- | --- |
| **Name** | **Module worked on** |
|  |  |
|  |  |
|  |  |
|  |  |